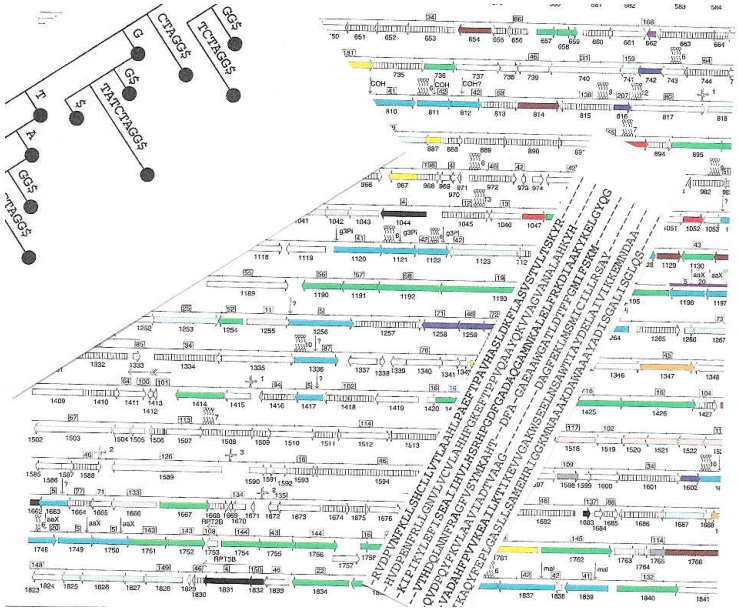
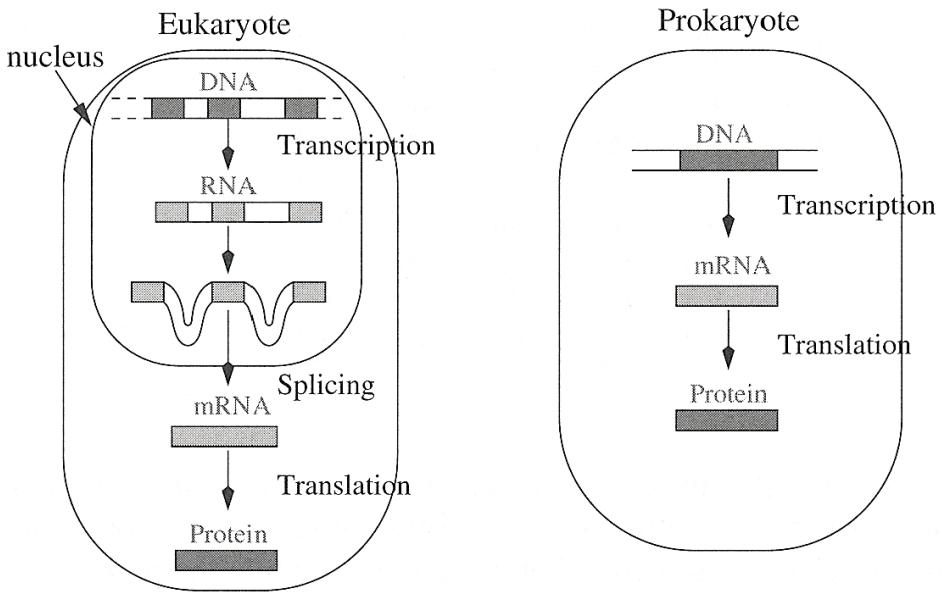
Lecture 7 : Gene finding

Principles of Computational Biology

Teresa Przytycka, PhD

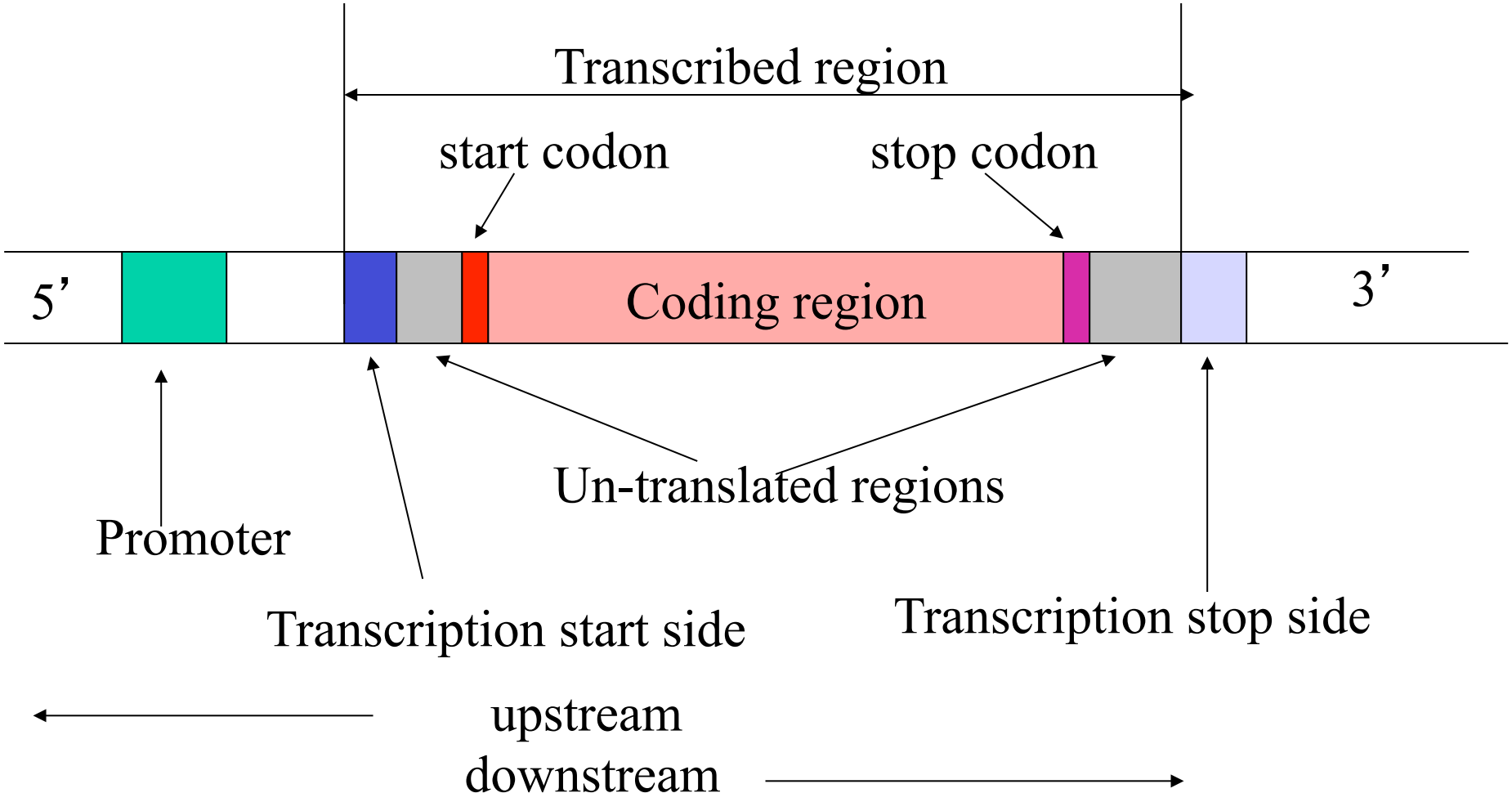


Transcription and translation in prokaryote and eukaryote



The different models of transcription and translation in prokaryotes and eukaryotes.

Transcription in prokaryote



-k denotes k^{th} base before transcription start, +k denotes k^{th} transcribed base

Issues in microbial gene finding

- Microbial genome tends to be gene reach (80%-90% of the sequence is coding)
- Major problem – which of two or more overlapping reading frames contains a gene (assumption – only one does)
- The most reliable method – homology searches (e.g. using BLAST and/or FASTA)
- Major problem – finding genes without known homologue cannot be based on such searches.

Method 1: Homology search for genes

- Translate the sequence in all six reading frames (3 forward and three reverse) and do a similarity search against the protein databanks. (BLASTX)
- Possible false positives - pseudogenes

Recognition of gene related signals

- Promoter side: bacteria TATAAT – Pribnow box at about –10; eukaryotes – TATA box at about –25.
- Termination: Could be done with or without help of the "rho" protein (rho dependent termination and rho independent termination). For rho-independent termination an inverted repeat sequence is required (to facilitate formation of a hairpin) .
- Other signal recognition –e.g binding motives
- Recognition method for such signals – motif search

Signal recognition for TATAAT

Recall positional weight matrix

position	1	2	3	4	5	6
A	2	95	26	59	51	1
C	9	2	14	13	20	3
G	10	1	16	15	13	0
T	79	3	44	13	17	96

f_A – expected frequency of A in genome

$f_{A^{i},i}$ – expected frequency of A in i^{th} position of Pribnow box

log-likelihood ratio:

$$\log\left(\frac{P(S|\ S \text{ is the box})}{P(S|\ \text{random sequence})}\right) = \log\left(\frac{\Pi_i f_{A^{i}i}}{\Pi_i f_A}\right) = \Sigma \log\left(\frac{f_{A^{i}i}}{f_A}\right)$$

TTATAAACATGCAAAATCCTCACTTATCAACAAAATGAACAATGTTTTAACAAAGTTTTG
AACAACAAAACCACATAAACTTCCCTCGAAAAACAAAGTTTTAAACAGTTTCCACACCCC
CTAAAGAAGAAGAGAATATTATGTATAAATATAACAATATATATTTAGGACCTGTGGAAA
CTGTTGAAAGGATCTGAGAAATGAGTTCTTTAACACTAAGTAGACGTCCTAGAAGAAACC
GAAAGACAGCAGCTATAAGAGATTTGTTGGCCGAAACTCACTTAAGCCCAAAAGATCTCA
TAGCACCATTCTTTGTGAAGTATGGAAATAACATAAAGGAAGAGATCCCGAGTCTTCCTG
GAGTGTTCCGATGGAGTTTGGATTTGCTATTAAAGGAAATAGAGCGTTTGTGTACCTACG
GGTTACGAGCTGTAATGCTGTTTCCCATTATTCCTGATGATCTTAAAGATGCTTACGGTT
CTTACTCCTCAAATCCTAAAAACATCTTATGTCATAGCATTCATGAAATAAAAAACGCAT
TTCCTCACCTATGTCTGATTAGTGATATAGCTTTAGATCCTTATACGACACACGGTCATG
ATGGGATTTTCCTTAATGGAGAGGTCCTTAATGATGAAAGTGTTAGAATTTTTGGAAATA

Method 2: Finding long ORFs

Open Reading Frame (ORF) is a sequence of codons which starts with start codon, ends with an end codon and has no end codons in-between.

Searching for ORFs – consider all 6 possible reading frames: 3 forward and 3 reverse

Does the ORF contain a gene ?

1. **Must be long enough (roughly 300 bp or more)**
2. **Should have average amino-acid composition specific for a given organism.**
3. **Should have codon use specific for the given organism.**

Motivation: Log enough sequence that does not code for a is expected to contain a stop codon

For a long enough ORF we can check codon composition

codon position	A	C	T	G
1	28%	33%	18%	21%
2	32%	16%	21%	32%
3	33%	15%	14%	38%
frequency in genome	31%	18%	19%	31%

Statistics collected from known genes

Genes tends to have codon composition characteristic for a given organism.

$$\frac{P(x|\text{in coding})}{P(x|\text{random})} = \prod_i \frac{P(Ai \text{ at ith position})}{P(Ai \text{ in the sequence})}$$

Score of AAAGAT:

$$\frac{.28*.32*.33*.21*.32*.14}{.31*.31*.31*.31*.31*.19}$$

More sophisticated method (Krogh)

Collect statistics about triples of nuclotides and for each triple x compute

$$\frac{P(x|\text{in coding frame})}{P(x|\text{random})} = \Pi_i \frac{P(abc|\ abc \text{ is a codon})}{P(abc|\ abc \text{ is random})}$$

Codon	Amino Acid	Usage	Random
AAA	Lys	3.5	1.3
AAG	Lys	1.1	1.6
AAC	Asn	2.4	1.4
AAT	Asn	1.4	1.3
⋮	⋮	⋮	⋮
GAT	Asp	3.2	1.5
⋮	⋮	⋮	⋮
TTT	Phe	1.9	1.2

Table 2: Frequencies (in percent) at which each of the 6

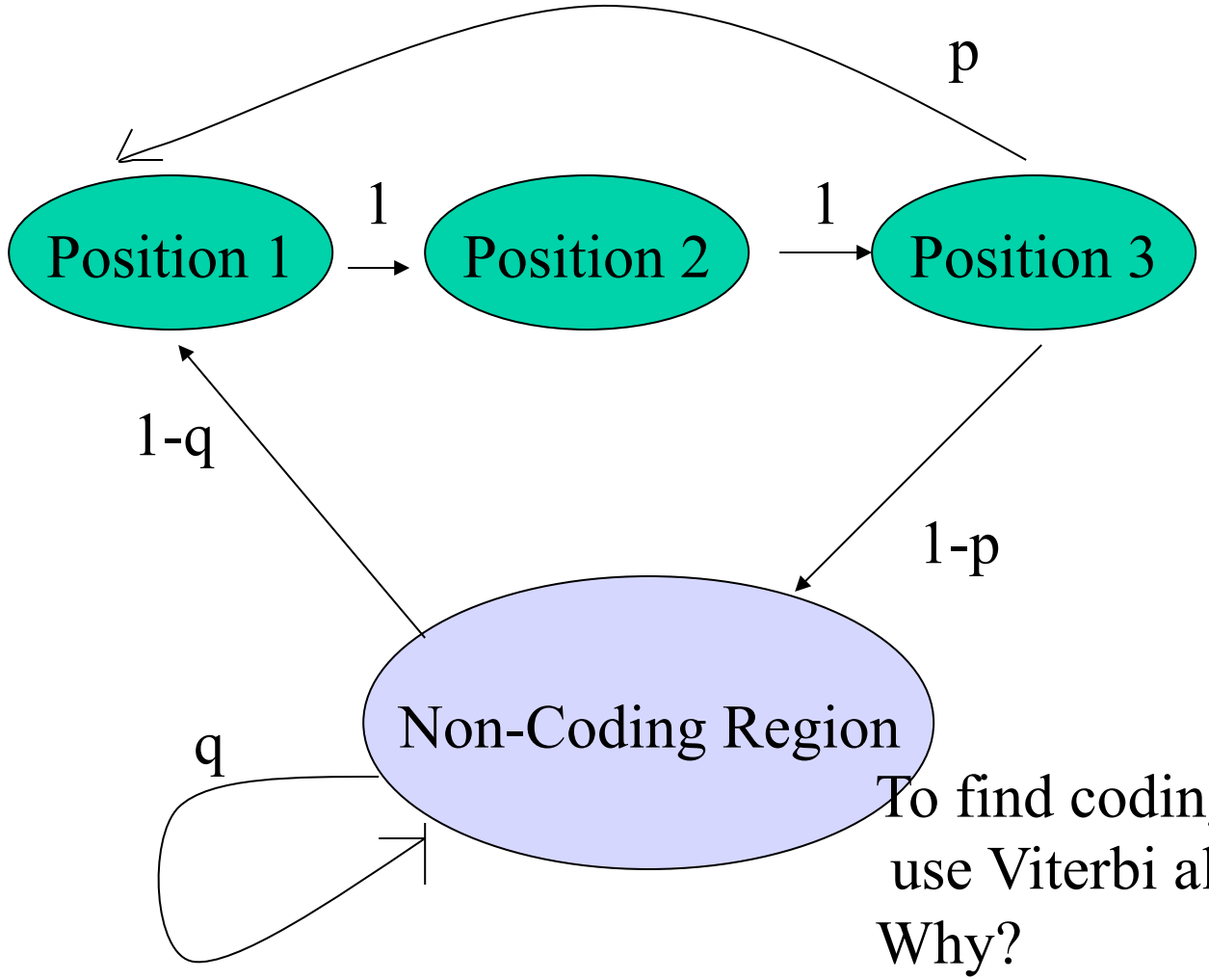
Table from ://www.stat.berkeley.edu/users/terry/Classes/s260.2000/

Let $p_{abc} = P(abc|\ abc \text{ is a codon})$

Score of AAAGAT:

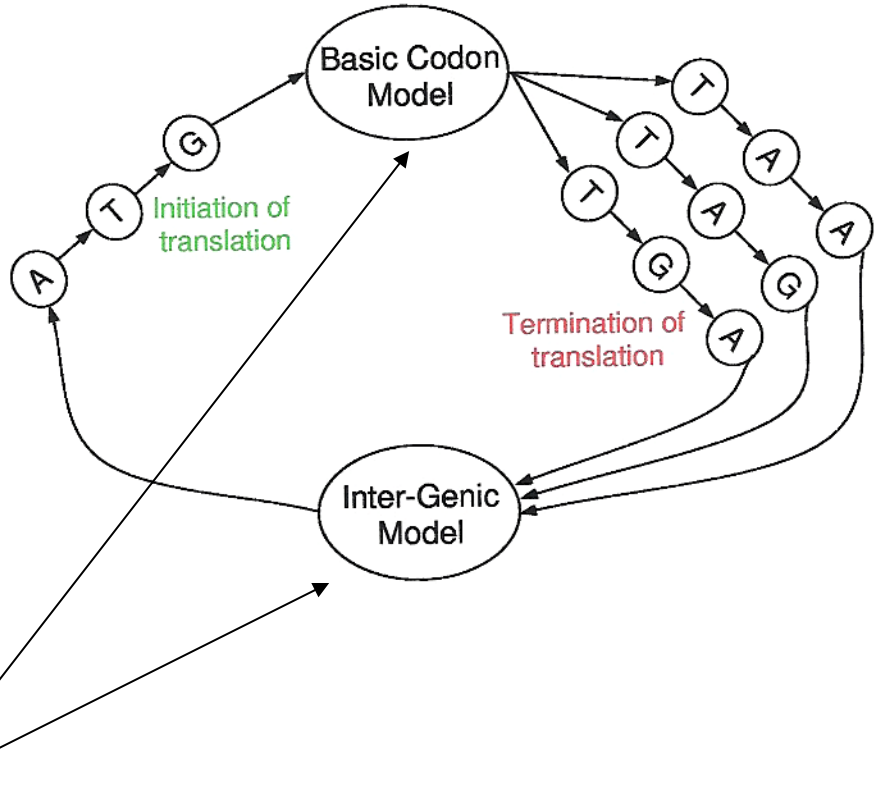
$$\frac{.035*.032}{.013*.015}$$

Method 3: HMM for prokaryotic genes



To find coding regions use Viterbi algorithm. Why?

A more general HMM for prokaryotic genes



these are usually HMM

www.stat.berkeley.edu/users/terry/Classes/s260.2000/

HMM-based Gene finding programs for microbial genes

- GenMark- [Borodovsky, Mcinnich – 1993, Comp. Chem., 17, 123-133] 5^{th} order HMM (requires estimating $4^{5+1} = 4096$ probabilities)
- GLIMMER[Salzberg, Delcher, Kasif,White 1998, Nucleic Acids Research, Vol 26, 2 55408] – Interpolated Markov Model (IMM) – method that avoids some of the problems related to insufficient data

Where the training data comes from

- Homology to known genes
- EST data
- Long ORF's

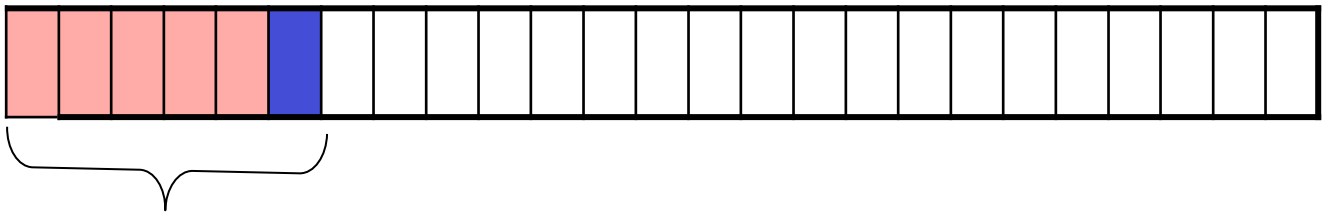
k^{th} order Markov Models

- Probability of each event depends on k previous events:

$$a(x|x_1 \ldots x_k)$$

- It requires k+1 dimensional array for transition probabilities is needed. If the states are A,C,G,T total size is 4^{k+1}
- In practice, these probabilities come from statistics and large k require lots of data.

Main idea behind GenMark



- Move a window of size 6 trough a known gene and collect statistics for transition matrix of dimension 6.
- From collected data compute probability of a given symbol (blue) providing that the it is preceded by given 5 symbols (pink)

Main idea behind Glimmer

- Frequently there is not enough data to support k-th order model in general, but but there may be some (k +1)-mers that occur frequently enough to be good predictors.
- In general, one would like to use highest order model
- Only models that are supported by sufficient data are helpful.
- Idea – use a combination of probabilities coming from models of various order with weights that depends on an estimation of the "quality" of the model (checked by the program)
  - is there enough data to support model of given order?
  - is there an advantage in moving to higher order model?

The Glimmer Program:

- Part 1: Given training set build Markov model
- Part 2: Identify putative genes in entire genome:
  - Identify all orfs longer than a threshold
  - Score each orf in each reading frame. Identify orfs with score above a threshold.
  - Examine orfs selected above for overlaps
  - Score overlapping regions in each frame separately to see which frame score the highest. Chose best gene candidate base on putative gene length, score of the overlap, and other information.

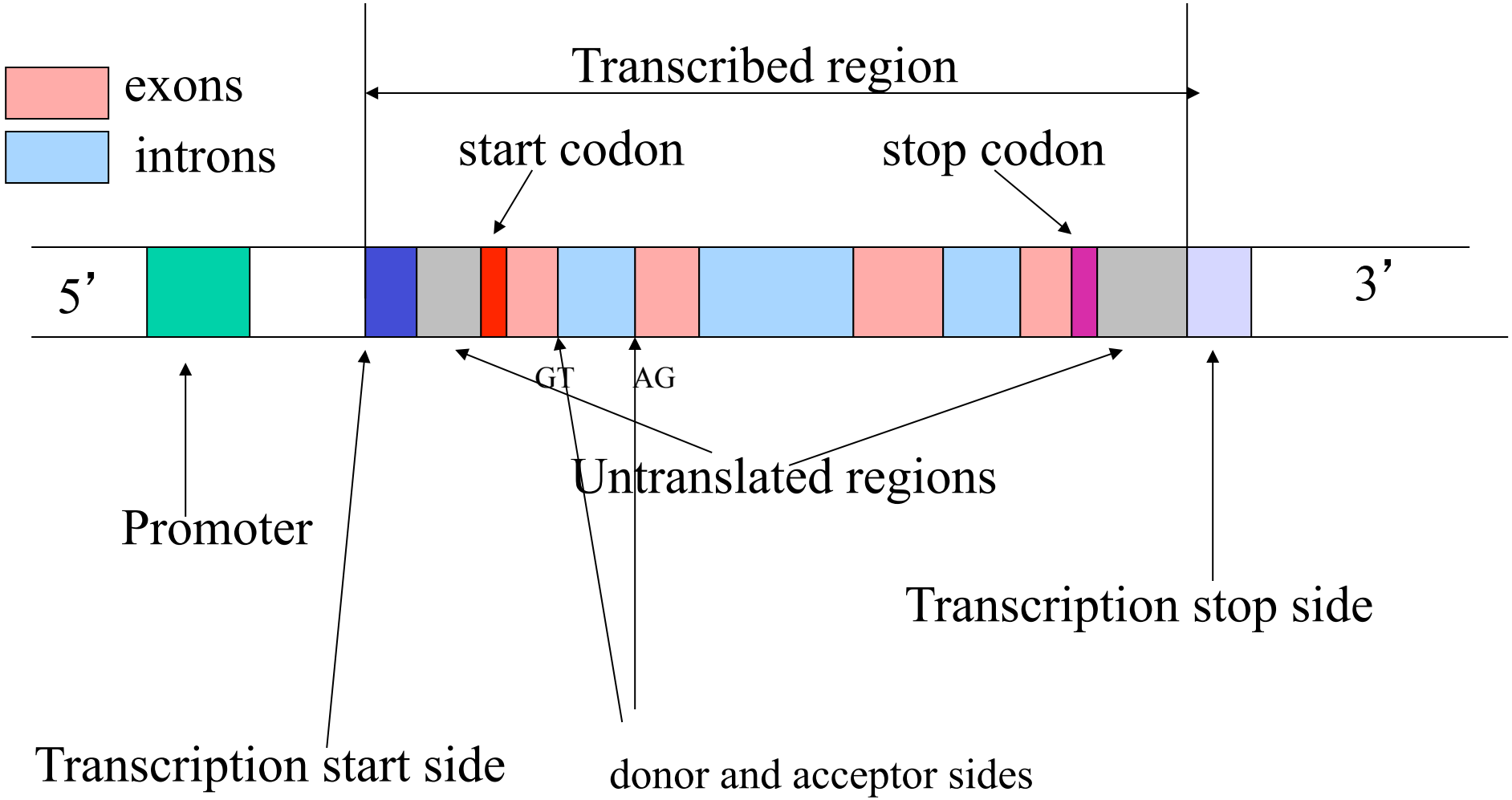
Comparison

Table 1. Comparison of the IMM model used in GLIMMER to a 5th-order Markov model

Model	Genes found	Genes missed	Additional genes
GLIMMER IMM	1680 (97.8%	37	209
5th-Order Markov	1574 (91.7%)	143	104

The first column indicates how many of the 1717 annotated genes in *H.influenzae* were found by each algorithm. The 'additional genes' column shows how many extra genes, not included in the 1717 annotated entries, were called genes by each method.

Gene structure in eukaryotes



Some statistics

- On average, vertebrate gene is about 30KB long
- Coding region takes about 1KB
- Exon sizes can vary from double digit numbers to kilo-bases
- An average 5’ UTR is about 750 bp
- An average 3’ UTR is about 450 bp but both can be much longer.

Methods

- Homology searches
- Dynamic programming
- HMM
- Generalized HMM
- Decision trees
- Computational linguistic
- Neural networks (like HMM, a machine learning method)
- …..

Homology searching

- By homology methods we can identify putative exons
- We need to align protein to a gene. But we need alignment methods that account for spliced introns – "spliced alignment"

Exon chaining problem

- Given: a set of putative exons such each exon has associated some weight
- Goal: Find the maximum weight non-overlapping sequence of exons

Information about exons may come from local alignment or BLAST searches and the weight may correspond to the score. It could come also from any exon prediction method

Form Jones and Pevzner book

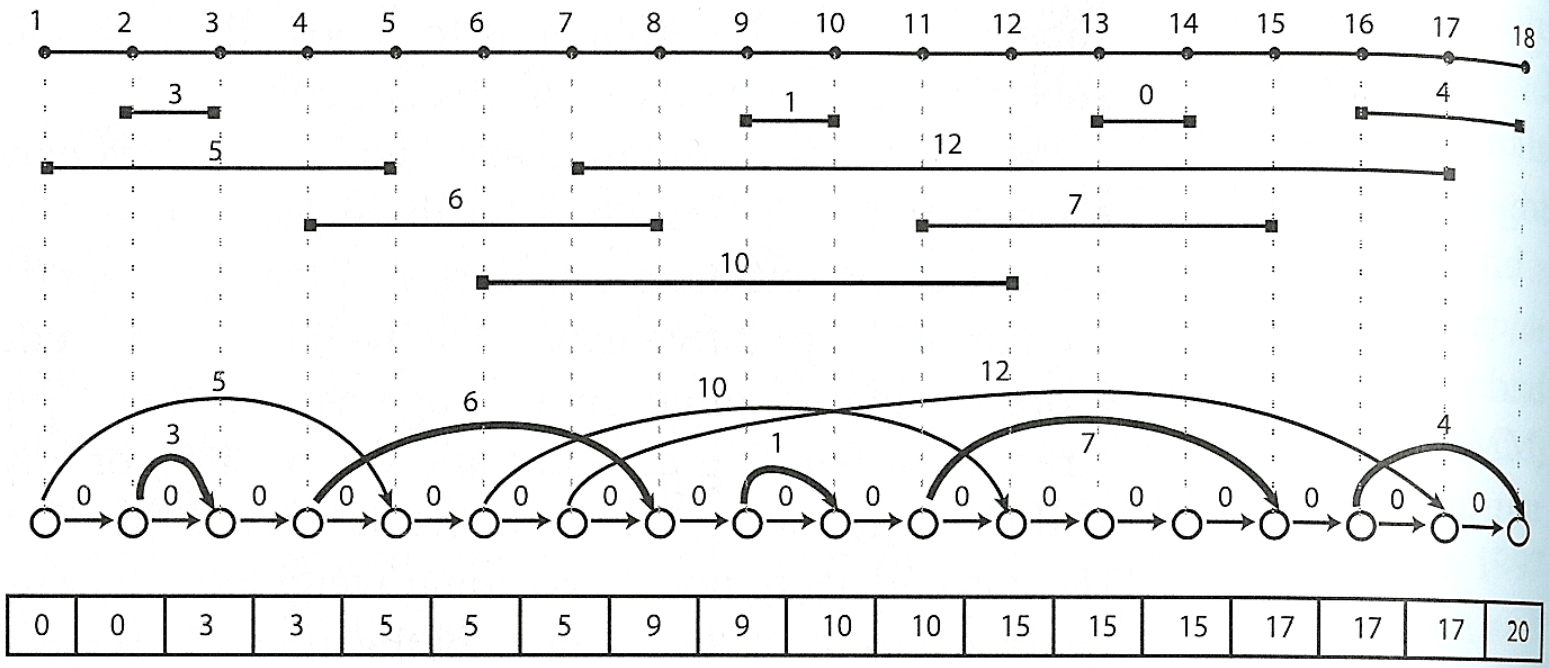


Figure 6.26 A short "genomic" sequence, a set of nine weighted intervals, and the graph used for the dynamic programming solution to the Exon Chaining problem. Five weighted intervals, $(2, 3, 3)$, $(4, 8, 6)$, $(9, 10, 1)$, $(11, 15, 7)$, and $(16, 18, 4)$, shown by bold edges, form an optimal solution to the Exon Chaining problem. The array at the bottom shows the values $s_1, s_2, \ldots, s_{2n}$ generated by the EXONCHANING algorithm.

Form Jones and Pevzner book

```
EXONCHAINING(G, n)
1  for i ← 1 to 2n
2      s_i ← 0
3  for i ← 1 to 2n
4      if vertex v_i in G corresponds to the right end of an interval
5          j ← index of vertex for left end of the interval I
6          w ← weight of the interval I
7          s_i ← max{s_j + w, s_{i-1}}
8      else
9          s_i ← s_{i-1}
10 return s_{2n}
```

Form Jones and Pevzner book

Spliced alignment

Spliced Alignment Problem:
Find a chain of candidate exons in a genomic sequence that best fits a target sequence.

Input: Genomic sequence G, target sequence T, and a set of candidate exons (blocks) $\mathcal{B}$.

Output: A chain of candidate exons Γ such that the global alignment score $s(\Gamma^*, T)$ is maximum among all chains of candidate exons from $\mathcal{B}$.

Method: Similar to chaining but you do "in parallel" two things: Chaining the interval and sequence alignment within each interval

Dynamic programming

GenPareser – Snyder and Strome 1993; in hybrid programs
FGENEH– Solovev et al 1995, MORGAN (Salzberg at al.)

- Let *i, j* positions in the sequence. Assume following scoring functions:
  - Intragenic(i,j) → can have two special subclasses – initial non-coding region and final non-coding regions
  - FirstExon(i,j) (must have start codon)
  - Exon(i,j)
  - Intron(i,j)
  - LastExon(i,j) (must end with end codon)
  - SingleExom(i,j) (must have both start and end codons)
- We assume, given a DNA sequence X we can score it as a possible candidate for any of the above regions

Parsing a gene – assigning optimal partition into block types

Let t be the type of block:

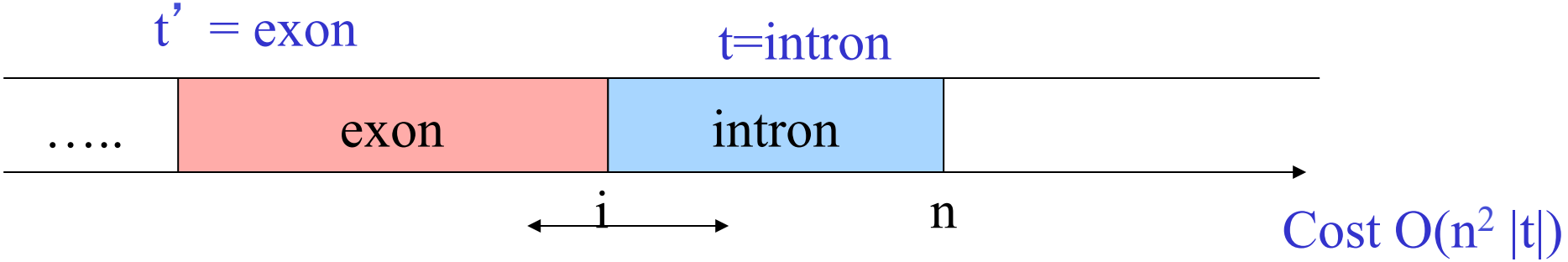
t = intron, first exon, middle exon,…

$S_t(i,j)$ the score of the sequence form i to j assuming the region t.

D[t,n] = score of the optimal partition of a prefix of X (first n letters) into blocks under assumption that last block of type t.

$D[t,n] = \max_{i<n,t'} (D[t', i]+S_t(i+1,n))$

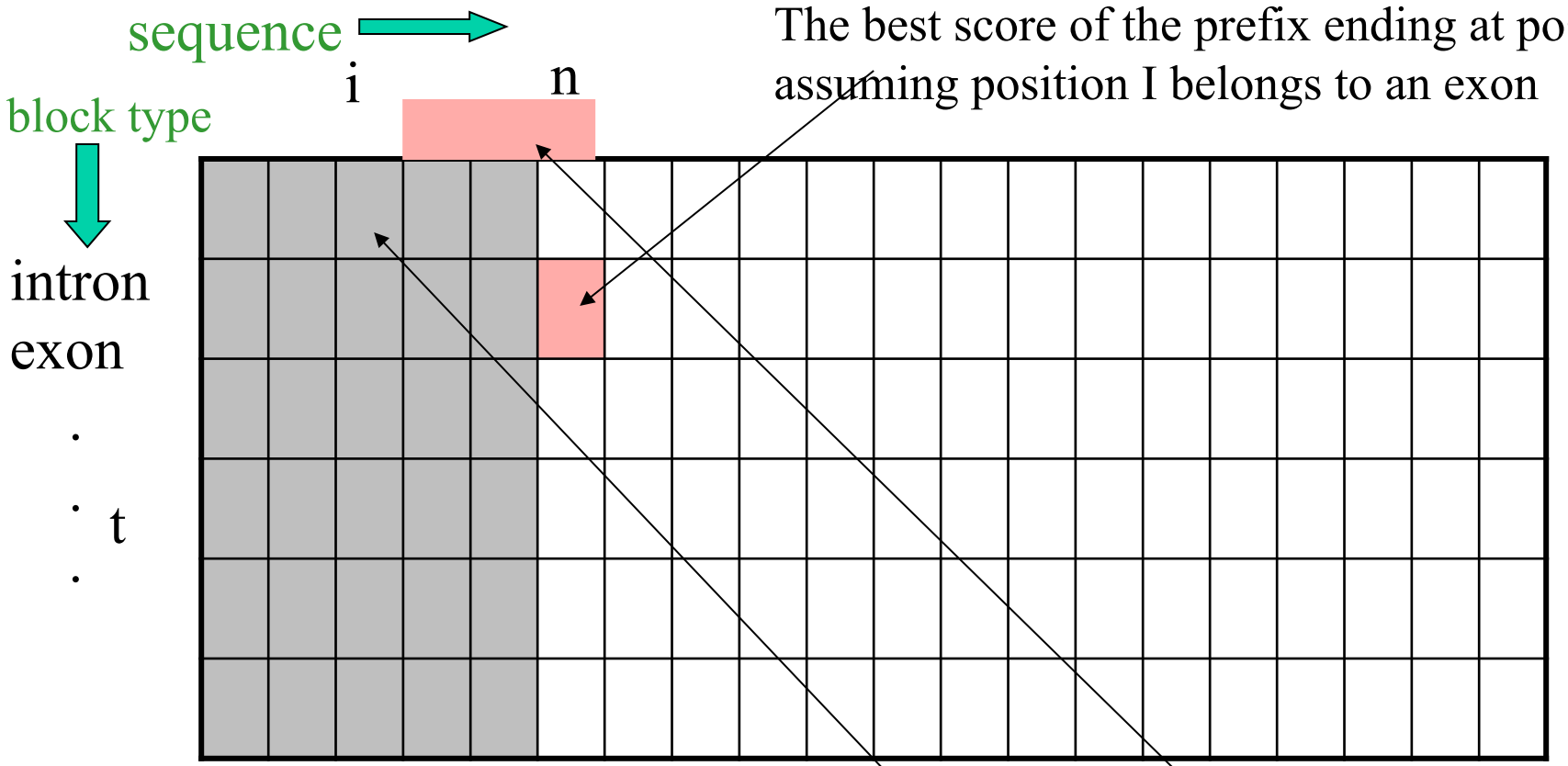
the maximum is taken over all possible t' of preceding blocks and all possible starting points i of the last block



Dynamic programming table D

D[exon,n]

The best score of the prefix ending at position i assuming position I belongs to an exon



Max is taken over gray region

$$D[t,n] = \max_{i<n,t'} (D[t', i] + S_t(i+1,n))$$

Max taken over the gray area

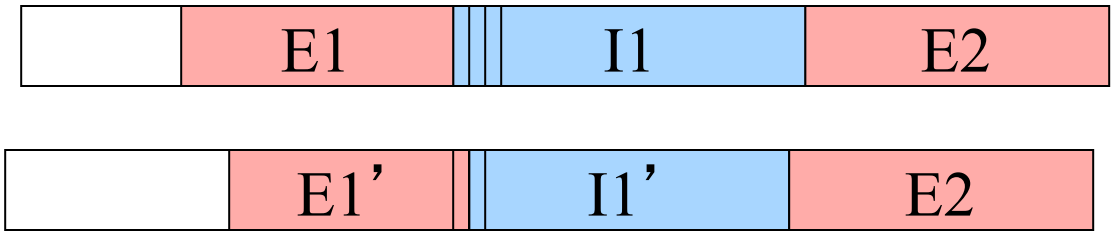
Exon score of segment from i+1 to n

Computing scoring function $S_t(i,j)$

- Collect statistics based on known genes (the composition of introns and exons, signal correlated with various blocks etc)
- For each nucleotide N, and each block type t (t=intron, exon,…), and position i in the block assign score of N belonging to the block t in a given position i.
- Possible scores:
  - log of probability (GeneParser)
  - HMM (Veil)
  - Decision tree model score (Veil)

Frame shift problem

As described the algorithm does not ensure frame consistency of frames within exons.



The score for E2 depends on what was the frame of E1'
When executing dynamic programming algorithm the decision which E1 I1 or E1' I1' is better depends on frame for E2.

For each location n we need to keep separate score for each frame

GeneParser

- Dynamic programming
- Authors:Snyder, E. E., Stormo, G. D. (1995) Identification of Coding Regions in Genomic DNA. *J. Mol. Biol.* **248**: 1-18.
- http://beagle.colorado.edu/~eesnyder/GeneParser.html

VAIL – a HMM based system

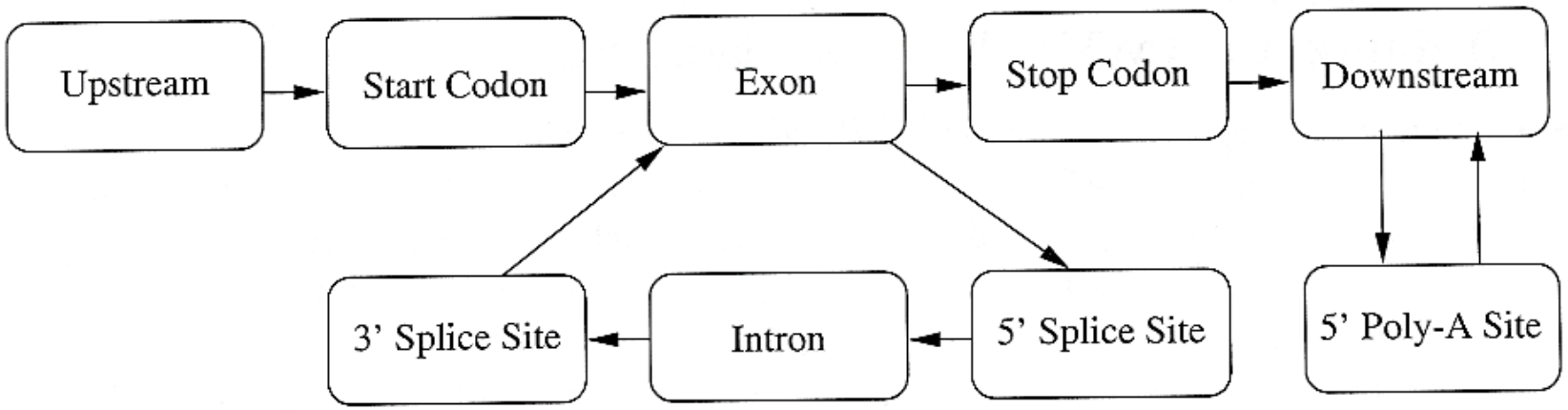
Henderson, Sazberg, Fasman *Journal of Computational Biology*, 4:2 (1997), 127-141

http://www.tigr.org/~salzberg/veil.html

Basic idea:

- Design a number of separate HMM models that capture properties of various regions (e.g. intron, exon)
- Train them separately.

Combined Model



Exon Model

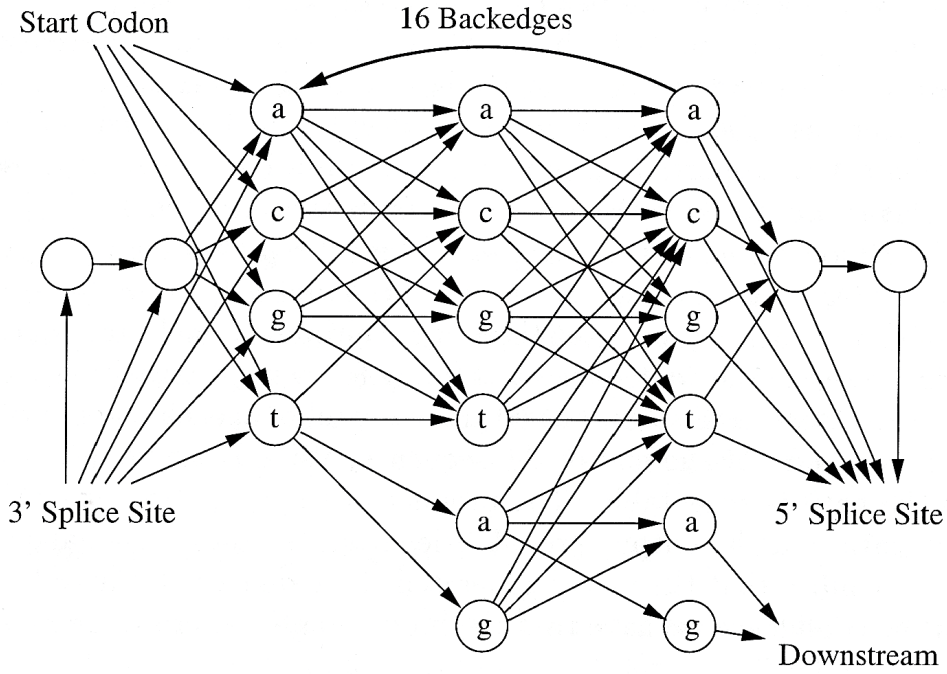


Figure 1: The exon and stop codon models in VEIL. This model can be entered in two ways: either just after outputting a start codon, or upon leaving the 3' splice site model, which follows the intron model. The three central columns of states correspond to the three codon positions. Each of these 12 states is labeled with the base that it can output. The system outputs bases three at a time, looping back after each codon. Note that the paths corresponding to a stop codon (TAA, TAG, and TGA) all force the system to exit from the model (four states at lower right of figure). Alternatively, the system can exit through the 5' splice site, in which case an intron must follow the exon. The two blank states on either end of the model can output any base; these "absorbing states" allow the model to align itself to the proper reading frame, as splice junctions need not respect codon boundaries.

Intron Model

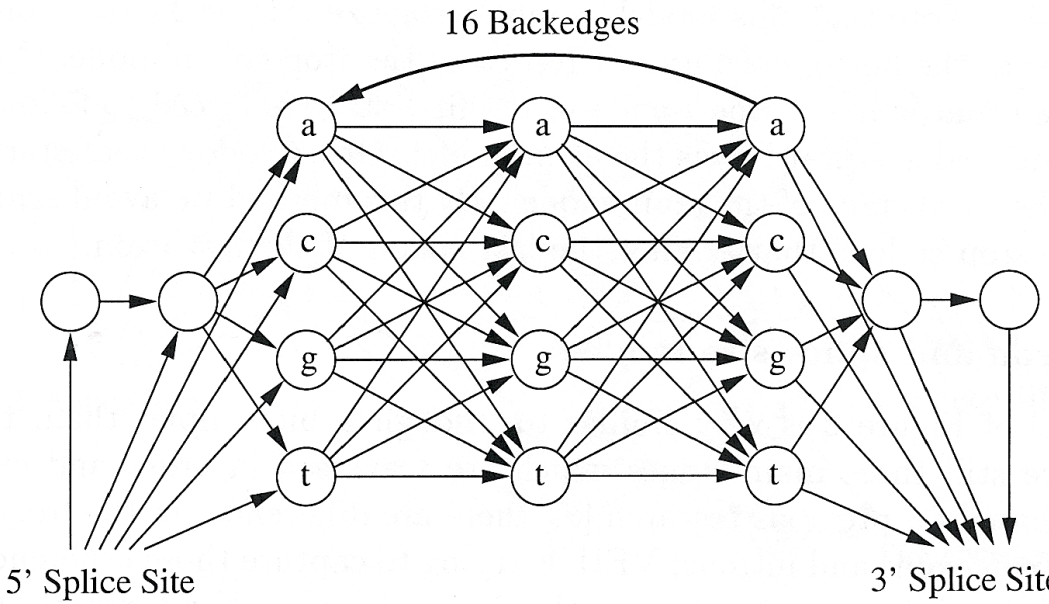


Figure 2: The intron model in VEIL. This model functions similarly to the exon model, with a few important differences. It reads bases three at a time in order to capture differences in the frequency of codon usage between coding and noncoding regions. Unlike the exon model, stop codons do not lead out of this model. The intron model must be entered and exited via splice junctions, which enforces the constraint that exons must appear on both sides of each intron.

Splice side models

There are statistics for consensus for donor and acceptor sides length (9 and 15 respectively). HMM corresponds to the consen

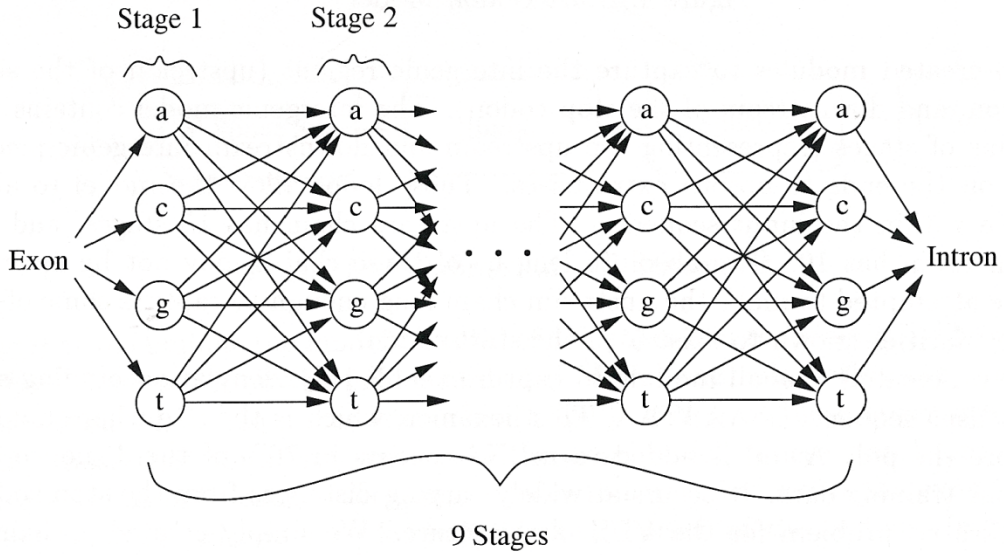


Figure 3: The donor site (5' splice site) model. Sequences must pass through this model to get from the exon model to the intron model. The exon-intron boundary occurs between stages 3 and 4; therefore stages 1-3 are part of the exon and stages 4-9 are part of the intron. Each state can output only one base, as indicated by the labels. Each edge between two states here contains the conditional probability of outputting a base in the latter state given the base shown in the previous state.

Donor side exon boundary: 3-4

Acceptor side – exon boundary 14-15

Training and parsing methods:

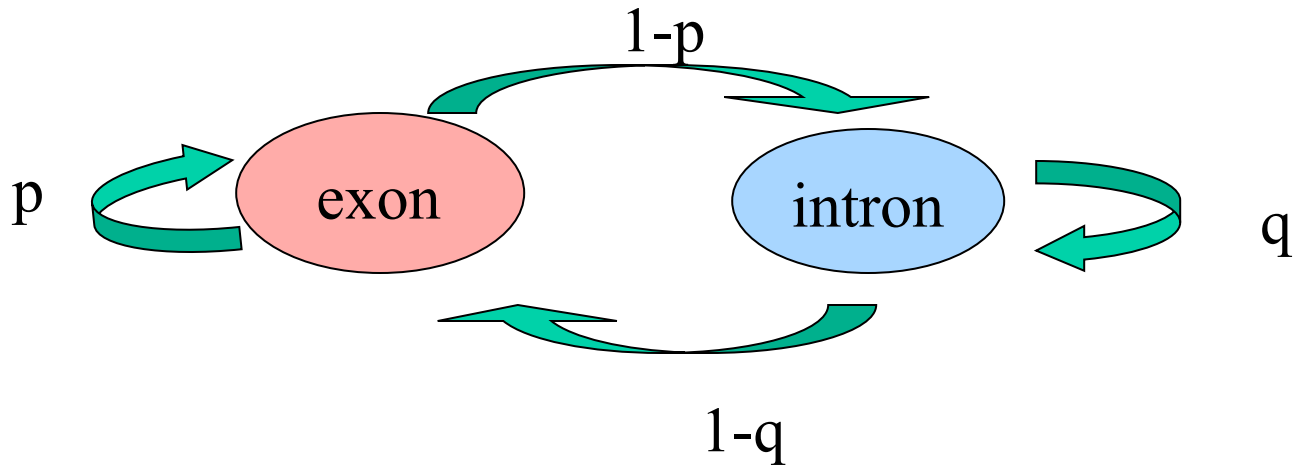
- Training: Expectation Maximization (EM)
- Parsing – Viterbi algorithm

GenScan program

- Uses so called generalized HMM where states generate whole sequences rather than single letters.
- States- correspond to different functional units of a genome (promoter region, intron, exon,….)
- States for introns and exons are subdivided according to "phase" three frames.
- There are two symmetric sub modules for forward and backward strands.

Performance: 80% exon detecting (but if a gene has more than one exon probability of detection decrease rapidly.

Generalized HMM



One thing that is not quite right with this HMM is that it models intron/exon length as if it had geometrical distribution. This is incorrect:
-very short exons are rear (spliceosme has hard time dealing with them)
-very long exons are rear too.

Generalized HMM addresses this problem as follows:

-At each state, rather then outputting a single symbol it outputs of a string of finite length

-the length of the output string string is randomly chosen with distribution that depends on state.

GENSCAN

Burge,Karlin, 1997